



# SystemC CCI WG Shared Parameters

P V S Phaneendra, CircuitSutra Technologies Pvt. Ltd  
May 2011



# Shared Parameters

- **Objective of the present example is to demonstrate the following:**
  - Sharing parameters among multiple modules

*Note: The approach shown herein uses C++ mechanisms as would be done with “plain old data types”. Example 11 shows a preferred alternative that takes better advantage of CCI Configuration capabilities.*

# Shared Parameters Sequence Diagram

## Within `sc_main`

Declare the instances of the owner and configurator classes

```
parameter_owner      param_owner("param_owner");  
parameter_configurator  param_cfgr("param_setter");
```

*CONFIGURATOR* will be made a *FRIEND* class to the *OWNER* class to access its private members

*set\_cfgr\_parameter* function returns the reference of the owner class parameters

```
param_cfgr.set_cfgr_parameter(&param_owner);
```

### Parameter\_Owner

Declare a *cci\_parameter*

```
cci::cci_param<int> int_param;
```

Assign default value to 5

Declare *CONFIGURATOR* as ***FRIEND*** class to the *OWNER*

```
friend class parameter_configurator;
```

### Parameter\_Configurator

Declare a *cci\_parameter*

```
cci::cci_param<int>* cfgr_shared_param;
```

Implements a function *set\_cfgr\_parameter* to get the reference of the owner parameter

```
void set_cfgr_parameter (parameter_owner *owner)  
{  
    cfgr_shared_param = &owner->int_param;  
}
```

# Cont'd

## Parameter\_Owner

@ 0 ns

Parameter Value : *int\_param.get\_value() : 10*

@ 5 ns

Set parameter value to 15  
*int\_param = 15;*

@ 20 ns

Parameter Value : *int\_param.get\_value() : 20*

## Parameter\_Configurator

Obtain a handle and set parameter value to 10 using it  
*cci::cci\_param\_handle cfgr\_param\_handle =  
m\_broker.get\_param\_handle("param\_owner.mutable\_int\_param");*

Change the value of the parameter to 10  
*cci\_param\_handle.set\_cci\_value(cci::cci\_value(10));*

@ 10 ns

Parameter Value : *cfgr\_shared\_param->get\_value() : 15*

@ 15 ns

Assign the parameter value to 20  
*\*cfgr\_shared\_param = 20;*

# NOTE: Up-front Planning Required

- **Accessing owner's parameters through shared parameters needs some up-front planning unlike the normal recommended way of doing it using base parameter objects:**
  - Configurator(setter) class is to be declared as a FRIEND to the OWNER class
  - Appropriate infrastructure must be created up-front to assign the references of the owner to the configurator.
- **P.S.: Not a recommended style of modeling cci applications. Use of shared parameters by the modeler is not appreciated.**

# Pros & Cons of Shared Parameters' Approach

## PROS:

- The performance of the cci-application will be relatively faster to the normal recommended approach of accessing owner's parameters using cci base parameter's objects, but modeler should be aware of the weird consequences that may arise.

## CONS:

- Since, the tool is gaining direct references to the owner's parameters, the tool will always track down the owner as the one who is responsible for the parameter value changes even though the value change is caused by the configurator(setter) class objects.
- Up-front planning needs to thought of first as listed within the earlier slide

# Expected Output (ex10\_Shared\_Parameters.log)

SystemC Simulation

Info: param\_owner: @0 s, [OWNER C\_TOR] : Default Value : 5

Info: param\_setter: @0 s, [CFGR C\_TOR] : Parameter exists

Info: param\_setter: @0 s, [CFGR C\_TOR] : Set parameter value to 10 via the handle

Info: sc\_main: Begin Simulation.

Info: param\_owner: @0 s, @ 0 s

Info: param\_owner: @0 s, [OWNER] : Parameter Value : 10

Info: param\_owner: @5 ns, @ 5 ns

Info: param\_owner: @5 ns, [OWNER] : Set parameter value to 15.

Info: param\_setter: @10 ns, @ 10 ns

Info: param\_setter: @10 ns, [CFGR] : Parameter Value : 15

Info: param\_setter: @15 ns, @ 15 ns

Info: param\_setter: @15 ns, [CFGR] : Change value to 20

Info: param\_owner: @20 ns, @ 20 ns

Info: param\_owner: @20 ns, [OWNER] : Parameter Value : 20

Info: sc\_main: End Simulation.